UNITED STATES DISTRICT COURT
NORTHERN DISTRICT OF NEW YORK

SUSAN B. LONG, DAVID BURNHAM, and TRAC REPORTS, INC. Plaintiffs, v. U.S. IMMIGRATION AND CUSTOMS ENFORCEMENT, and U.S. CUSTOMS AND BORDER PROTECTION, Defendants.	Civil Action No.: 5:23-cv-01564 (DNH/TWD)

DECLARATION OF TIMOTHY GIBNEY

Timothy Gibney, declares the following under the penalties of perjury:

1. I am a Supervisory Management and Program Analyst within Enforcement and Removal Operations ("ERO"), Law Enforcement Systems Analysis ("LESA"), Information Technology Management ("ITM") Unit, at U.S. Immigration and Customs Enforcement ("ICE"). I have held this position since August 2017. I oversee system development activities related to ERO data initiatives. This includes the management of data warehouses and reporting tools. Prior to my employment at ICE, I worked for several consulting firms supporting several government clients within the Department of Justice, United States Postal Service, and the Department of Homeland Security ("DHS"). My 17 years of consulting experience prior to federal employment was primarily focused on Information Technology ("IT") system requirement definition, design, development, implementation, training, and maintenance. This includes approximately seven years directly supporting ICE and the applications and databases that support ERO operations.

2. LESA is responsible for providing operational reports and strategic analysis to stakeholders and senior leadership on behalf of ERO, and providing all official ERO reporting to Congress, the White House, and other internal and external stakeholders, as well as providing information to the public through the FOIA. LESA delivers analytical tools, studies, reports, IT business requirements, and process-improvement recommendations through data collection and technology integration. LESA leads ERO's efforts towards modernization and organizational transformation. In addition, LESA supports system deployment training, communications, and user adoption efforts. This includes the development of strategic plans prior to a system deployment or release and the delivery of training and technical support throughout the deployment.

3. LESA also provides oversight for all data quality and integrity efforts to improve the reliability of immigration enforcement and removal data in systems used by ICE. LESA supports ERO by ensuring that data is correctly entered in ICE systems, information is compliant with policy and best practices, and that data quality assessment and measurement standards are applied throughout the ERO immigration lifecycle.

4. LESA provides strategic operations and analysis support to inform management decisions at the ICE Headquarters and Field levels. LESA ensures the accountability, consistency, and efficiency in the statistical reporting of ERO operations and establishes and maintains standard methodologies and routinely reports on key performance metrics such as detainers, arrests, book ins, removal, and national dockets. LESA provides weekly, bi-weekly, monthly, quarterly, and annual routine reports for DHS components and other government stakeholders and provides ad hoc analysis and reporting on a regular basis to support requests that include congressional inquiries, inter-agency data, litigation, and FOIA requests.

5. I make this declaration in support of ICE's motion for summary judgment in this action. The statements contained in this declaration are based upon my personal knowledge, training, and experience as well as my review of documents kept by ICE in the ordinary course of business, and information provided to me by other ICE employees in the course of my official duties.

PLAINTIFFS' CURRENT FOIA REQUEST TO ICE

6. I have reviewed an August 1, 2024, letter from Nick Sansone, Esq., to David M. Katz, Esq., which I have been told constitutes Plaintiffs' request for records from ICE for the purposes of this motion.

7. In the letter, Plaintiffs requested "[a]ll datapoints (from any time) that are directly or indirectly linked to a person for whom the Agencies have established an official case seeking that person's removal from the country" from the [Enforcement Integrated Database] "EID." Based on my training, knowledge, and experience, I understand this request as asking ICE to produce all EID data pertaining to any person who Customs and Border Protection "CBP" or ICE has "established an official case seeking that person's removal from the country." This request is not limited to a specific time period. Therefore, for illustration purposes only, if ICE "established an official case seeking" a certain person's removal in 2014, Plaintiffs are requesting every piece of data contained in the EID related to that individual. This request could include, for example, data about the person from 2004 as well as data about the person from 2024. The EID is not structured in this way, so (as explained below), an in-depth analysis and computer coding effort would be needed to allow ICE to extract this data from the EID. Plaintiffs' request also includes information directly or indirectly linked to the person. I will refer to this as "Part 1" of the request.

8. In the same letter, Plaintiffs further request "[a]ll datapoints (from any time) that are directly or indirectly linked to a person who was apprehended pursuant to a CBP 'encounter' in or after Fiscal Year 2020." 'Encounter' is used to mean the same thing that CBP uses it to mean in its Nationwide Encounters Dataset[1]" from the EID. I will refer to this as "Part 2" of the request. While Part 2 of the request appears to have a temporal limitation at first glance (pertaining to encounters from Fiscal Year 2020 forward), in fact Plaintiffs still seek data "from any time" pertaining to those people. Further, while the number of people involved in Part 2 may be smaller than in Part 1, searching for data pertaining to those people has no time limit.

9. I understand that Plaintiffs seek the ability to preserve "relational information" and "linkage fields" as well as potentially substitute identifiers so that Plaintiffs can track data that relates to the same individual even if free-format fields, meaning fields that contain unique individualized content such as names, dates of birth, addresses and the like, are redacted.

10. I have reviewed the Plaintiffs' request for data to be sourced directly from EID for "[a]ll code files, lookup tables, or other records that translate the specific codes used in connection with the datapoints contained in paragraphs (1) and (2) above into their corresponding meaning." I interpret this as asking us to ensure that any "code" value either be directly translated into a description within the results of the Part 1 and Part 2 searches or be provided as stand-alone code lookup tables which would allow the plaintiff to manually translate the codes. For example, if a row of data in the results for Part 1 or Part 2 included a code value of "VA", we could either translate that to "Virginia" in a data field directly in the Part 1 and 2 results, or provide a separate

[1] While I do not speak for CBP, it is my current understanding that CBP encounter could mean any encounter of a removable noncitizen by CBP, Office of Field Operations (OFO) or U.S. Border Patrol (USBP), including the arrest of a removable noncitizens by USBP under Title 8 authority, a determination of inadmissibility for a person requesting admission at a port of entry, or Title 42 expulsions, for example.

table which translates all state codes to their corresponding state descriptions which the Plaintiffs would use to manually translate "VA" to "Virginia". I will refer to this as "Part 3" of the request.

BACKGROUND ON EID DATABASE STRUCTURE

11. Plaintiffs seek EID data that is "directly or indirectly" linked to two sets of people. The requested data is dispersed diffusely throughout many parts of the EID, and it is not kept or maintained in a way that would allow ICE to simply search for the data using existing capabilities. To the contrary, ICE will have to engage in significant work to determine how to search for responsive EID data (and to determine which EID data should be included in the search) and then will have to engage in significant work to create a means to search for such information. To understand the burdens associated with searching for records responsive to Plaintiffs' request, it becomes necessary to understand some basic concepts of database structure and how those concepts relate to the EID.

12. The EID database is owned, operated, and managed by ICE. It stores and maintains information related to the investigation, arrest, booking, detention, and removal of persons encountered during immigration and criminal law enforcement investigations and operations conducted by ICE, U.S. Citizenship and Immigration Services ("USCIS"), and CBP. The EID is the common database repository for all records created, updated, and accessed by a number of software applications, including several DHS law enforcement and homeland security applications.

13. The EID is a large database containing millions of records which are stored in over 1,000 data tables and over 12,000 unique data fields.

14. At its core, like any database, the EID is designed to house and organize data. In databases like the EID, individual datapoints are held in fields within data tables. If we consider a hypothetical example from the commercial arena, a database might include a table for orders a business receives and another table for its customers. Here are two simple tables:

TABLE:
ORDERS

ORDER ID	**ORDER DATE**	**ORDER TIME**
2024-01	*01/02/2024*	*9:00 a.m.*
2024-02	*01/03/2024*	*5:00 p.m.*
2024-03	*1/04/2024*	*7:20 a.m.*

TABLE:
CUSTOMERS

CUSTOMER ID	**CUSTOMER NAME**
0001	*ACME*
0002	*GenCo*
0003	*ABC Co*

15. The "Orders" table allows each record to contain three possible data fields, which are bolded: (1) Order ID; (2) Order Date; and (3) Order Time. The table contains three records, which in this example are the rows data fields. Each record has three data points.

16. The "Customers" table allows each record to contain two possible data fields: (1) Customer ID; and (2) Customer Name.

THE EID'S STRUCTURE AS A TRANSACTIONAL DATABASE

17. There are several types of databases. The EID is a transactional database. A transactional database is designed to handle frequent transactions—records of actions or activities—like buying something online or updating account details.

18. For example, businesses use transactional databases for tasks like processing orders, managing inventory, or handling customer information. Transactional databases ensure that each transaction or record is completed accurately and reliably, following rules that maintain data integrity. For example, rules would be in place that require that an order be linked to a customer, but not all customers may be linked to an order.

19. The EID stores transactions relating to the immigration enforcement lifecycle, such as an arrest, release from a detention facility, a deportation or granting of immigration benefits. Transactional databases, like the EID, are primarily designed to allow efficient access by users for high-speed, real-time operations that involve frequent updates, inserts, and deletions. It also contains law enforcement sensitive information relating to investigations, enforcement operations, and checks of other law enforcement databases. The EID is a 24x7 operational database with more than 55,000 users. While transactional databases excel at handling individual transactions efficiently, they are not optimized for analytics and reporting. Extracting a large dataset for analytical purposes from the EID would be inefficient and lead to long query times and high resource consumption. As a result, attempting complex queries or large-scale data aggregation in the EID can be challenging and may not deliver the performance needed for effective analysis.[2]

[2] A data warehouse, in contrast to a transactional database, is designed to gather and store large amounts of historical data from source databases, which are often transactional. Data warehouses organize data in a way that makes it easy to run complex analyses and generate reports. Instead of handling transactions, a data warehouse is designed for understanding trends and patterns over time, making them ideal for analytics, reporting, and business intelligence. ICE maintains the ICE Integrated Decision Support ("IIDS") data warehouse which sources relevant data from the EID to support ERO's data reporting and analytics functions.

20. As a transactional database, the EID uses a design concept called database normalization, which means EID data is organized into over 1,000 smaller tables of data. In a normalized database, the design aims to have single piece of data (like the name of an individual) "reside" in one singular field. These independent "addresses" for data are known as a "primary key," which is a unique identifier for each record in a database table.

21. While the piece of data may directly or indirectly relate to records in several tables, the piece of data only exists at one metaphorical "address." Thus, the data is "linked" so that what the user sees is the original "address" instead of being entered in multiple spots in the database.[3] For example, instead of repeating customer details in every order, a transactional database might have one table for customers and another for orders, linking them together by means of a "foreign key," which is a field in one table that links to the primary key in another table. A foreign key is used to create a direct relationship between two tables. Here are the "Orders" and "Customers" tables with an additional data field to link the tables using a foreign key (denoted "FK" below) that links back to data at the primary key (denoted "PK" below):

[3] One way to think about this concept is live radio or television. When Americans listen to a live speech or watch a live interview, for example, those listening on the radio to the speech or watching on television are hearing or seeing a live event in potentially millions of homes and businesses throughout the nation. But the person delivering the speech is only in one place, as is the interviewer and the interviewee. Thus, even though thousands of people are listening or watching from different places, they are all seeing a singular event. So, too, while a user may "see" the same data displayed multiple times in a user interface, what the user is seeing is in effect a broadcast of the data from its actual location as opposed to seeing the same data in a different place.

TABLE:
ORDERS

ORDER ID (PK)	ORDER DATE.	ORDER TIME	CUSTOMER ID (FK)
2024-01	*01/02/2024*	*9:00 a.m.*	*0001*
2024-02	*01/03/2024*	*5:00 p.m.*	*0002*
2024-03	*1/04/2024*	*7:20 a.m.*	*0001*

TABLE:
CUSTOMERS

CUSTOMER ID (PK)	CUSTOMER NAME
0001	*ACME*
0002	*GenCo*
0003	*ABC Co*

22. When a user wants to review what orders have come in using their order management software, a user might see the following on the screen (via a "user interface"):

USER INTERFACE:
ORDERS WITH CUSTOMERS

ORDER ID	ORDER DATE	ORDER TIME	CUSTOMER ID	CUSTOMER NAME
2024-01	*01/02/2024*	*9:00 a.m.*	*0001*	*ACME*
2024-02	*01/03/2024*	*5:00 p.m.*	*0002*	*GenCo.*
2024-03	*1/04/2024*	*7:20 a.m.*	*0001*	*ACME*

23. Even though the user "sees" all five fields for each order record in a single table on their screen, the user is not seeing any new *data* and the data is not stored in a single table in the database. When a software application presents data to an end-user from a transactional database, it first asks the database for specific information, like customer orders. The database then searches through its records to find the relevant data and sends the results back to the application. The application then shows this information in a user-friendly format, such as lists or charts, making it easy for users to see and interact with the data. The application acts as a bridge, helping users access and view the data they need efficiently.

24. Instead, the user is seeing a rendering of the data created by the application by sourcing data from the two tables of transactional data. The links back to the underlying database tables are explained below:

LINKS BEHIND USER INTERFACE:
COMBINED TABLE VIEWED IN ORDER APPLICATION

ORDER ID	ORDER DATE	ORDER TIME	CUSTOMER ID	CUSTOMER NAME
linked to "Order ID" in Orders Table	linked to "Order Date" in Orders Table	linked to "Order Date" in Orders Table	linked to "Customer ID" in Customers Table Joined to "Customer ID in the Orders Table	linked to "Customer ID" in Customers Table

25. This structure helps maintain data integrity and allows for quick updates, but it means analytical queries can be very complex and slow to process because, in colloquial terms, the data must be "constructed" to ensure the proper linked data appears in the output table.[4] This process of construction is what we refer to as a database query. Database queries are created through a structured programming language to retrieve the data stored, often from multiple tables, and produce the results in a single table for consumption. Queries can be very complex given the size and scope and the number of tables involved and how the tables are related. Also, the query may include calculations which enrich the data for consumption.

[4] Data warehouses, in contrast, are focused on analysis rather than speed for individual transactions. They use denormalization, which means combining related data into fewer, larger tables (the opposite of a transactional database design). This might mean storing customer details alongside their order history in one table. While this results in data duplication, it makes it much easier and faster to run complex queries and analyze trends across large datasets. In the example above, the combined Orders with Customers table which was viewed in the Order Application may exist as a single denormalized table in a data warehouse which would *actually contain the same data contained from the Orders table and the Customers table with Customer data repeated over and over for each order that customer placed*.

26. To explain this example in more technical terms, in a transactional database used to store customer and order information a table called "Customers" may exist with a primary key of Customer ID. Direct relationships would be instances where other tables have Customer ID stored in them as a foreign key which would establish a direct relationship. For example, in the "Orders" table, there will be a primary key named Order ID. And, in the "Customers" table, there will be a primary key named "Customer ID."

27. In a transactional database, like EID, since there are many tables, there are no direct relationships between all tables, meaning not every table has a foreign key to directly link it to every other table in the database. Using the Customer/Order example above, consider two more tables: "Products" and "Suppliers".

28. The product table will hold data about different products which may be ordered, and the supplier table will hold data about the external suppliers of those products. In this scenario, the "Orders" table will have a foreign key to the "Products" table, and the products table will have a foreign key to the "Suppliers" table.

29. Another topic of note when evaluating relationships between tables in a database is database cardinality. Cardinality refers to the relationship between tables and how many records in one table correspond to records in another. Key types include:

a. One-to-One: Each record in one table matches exactly one record in another. For example, each order has one unique payment transaction, and each payment transaction only relates to one order.

b. One-to-Many: A single record in one table can relate to multiple records in another. For instance, each customer may have many orders, but each order only relates to one customer.

c. Many-to-Many: Records in one table can relate to multiple records in another, and vice versa. For example, a single product may be supplied by multiple suppliers, and each supplier may supply multiple products. A best practice when designing a transactional database is to use link tables to resolve many-to-many relationships. The link table will contain two sets of foreign keys each linking back to the primary keys of the tables with the many-to-many relationship. So, in our product / supplier example, there will be a Product Suppliers link table in between the Products table and Suppliers table. Since one-to-many relationships frequently exist, link tables are very common.

30. These variations in cardinality further complicate the ability to conduct a search that includes data "directly or indirectly" linked and that preserves "relational information" to data.

31. Consider a requirement to query which customers may have received products from specific suppliers so that customers can be notified about a product recall. To do this, the person searching the database would attempt to establish a relationship between the "Customers" and "Suppliers" tables. There is no direct primary/foreign key relationship between these two tables, so an indirect relationship would need to be established. In this scenario, an indirect relationship does exist because we can join "Customers" to "Orders," then "Orders" to "Products," and then "Products" to "Suppliers" using three different foreign keys and link tables. Thus, even in a fairly simple example, the database design can be quite complex. Also this example, like EID, not all records in one table have links to records in other tables. For example, we have a customer, ABC Co which has never placed an order so has no direct or indirect links to any data elements in other tables. Likewise, we have a supplier setup, 1-2-3 Industries, which has yet to supply any products, so they have no direct or indirect links to any other records.

TABLE:
CUSTOMERS

CUSTOMER ID (PK)	CUSTOMER NAME
0001	*ACME*
0002	*GenCo*
0003	*ABC Co*

TABLE:
ORDERS

ORDER ID (PK)	ORDER DATE.	ORDER TIME	CUSTOMER ID (FK)
2024-01	*01/02/2024*	*9:00 a.m.*	*0001*
2024-02	*01/03/2024*	*5:00 p.m.*	*0002*
2024-03	*1/04/2024*	*7:20 a.m.*	*0001*

TABLE:
PRODUCT ORDERS

ORDER ID (FK)	PRODUCT ID (FK).	QTY	SIZE
2024-01	*P-001*	*1*	*XL*
2024-02	*P-002*	*4*	*M*
2024-02	*P-003*	*1*	*34-30*
2024-03	*P-002*	*5*	*S*

TABLE:
PRODUCTS

PRODUCT ID (PK)	PRODUCT DESC
P-0001	*T-Shirt*
P-0002	*Polo Shirt*
P-0003	*Pants*

TABLE:
PRODUCT SUPPLIERS

SUPPLIER ID (FK)	PRODUCT ID (FK).
S-0001	*P-001*
S-0001	*P-002*
S-0002	*P-002*
S-0003	*P-003*

TABLE:
SUPPLIERS

SUPPLIER ID (PK)	SUPPLIER NAME
S-0001	*XYZ Clothier*
S-0002	*Brand Name Garments*
S-0003	*ACME Slacks*
S-0004	*1-2-3 Industries*

32. Plaintiffs' request seeks to require ICE to search for data that is directly and indirectly linked, and to preserve the linkage where there is a link. Further, with indirect links, any search that will preserve these links will require significant analysis and computer coding.

33. Data entered into a transactional database is typically done through a user interface that interacts with the database. These user interfaces can be web-based applications or standalone software. During data entry, the user interface often enforces rules to maintain data integrity. Commonly entered values are stored in code lookup tables within the transactional database. For instance, when a user orders a t-shirt, they select the size from a dropdown menu rather than typing it into a free text field. To manage this, the database includes a size-codes table linked to the products table, ensuring that only valid size options are available to be selected for each product. The selected size code is then recorded in the orders table for the specific order. Sometimes not all valid inputs are known or defined so free-text input fields are used. An example of this would be a "Special Delivery Instructions" entry field on an order where the customer can enter free text data.

SEARCHING DATABASES LIKE THE EID REQUIRES USING QUERIES, WHICH INVOLVE COMPUTER CODING

34. The EID search required to comply with Part 1 and Part 2 of Plaintiffs' request would require ICE to first determine what information must be searched and then to write a query so that the database will "return" the data. A database query is a structured request for data from a database. It typically uses a language like the Structured Query Language ("SQL").[5] to specify what information is to be retrieved from the database and how to join data together from different

[5] SQL is a widely used standard programming language designed for managing relational databases. It enables users to extract and manipulate large volumes of data, presenting the results in a tabular format.

tables in the database. Anytime data is to be extracted from a database, a query must be developed and executed. In general, queries which are designed to extract data from a transactional database are much more complex than a query designed to extract data from a data warehouse.

35. Using our previous example of the database used to capture customer and order data, I will provide two sample queries using SQL code. The first example is fairly simple and shows how we would write a query to extract data on customers and orders, which have a direct link:

a. SELECT

b. ORDERS.ORDER_ID,

c. ORDERS.ORDER_DATE,

d. ORDERS.ORDER_TIME,

e. CUSTOMERS.CUSTOMER_ID,

f. CSUTOMERS.CUSTOMER_NAME

g. FROM ORDERS

h. INNER JOIN CUSTOMERS ON CUSTOMERS.CUSTOMER_ID = ORDERS.CUSTOMER_ID

36. In the next example, I will demonstrate joining data together from the Customers and Suppliers tables, which have only an indirect link. The SQL code gets more complex. In this example, we need to make 5 joins to merge data from the two tables. If performing a query that is joining hundreds of tables together, this code would get exponentially more complex:

a. SELECT

b. CUSTOMERS.CUSTOMER_ID,

c. CUSTOMERS.CUSTOMER_NAME,

d. ORDERS.ORDER_ID,

e. ORDERS.ORDER_DATE,

f. ORDERS.ORDER_TIME,

g. PRODUCTS.PRODUCT_ID,

h. PRODUCTS.PRODUCT_DESC,

i. PRODUCT_ORDERS.QTY,

j. PRODUCT_ORDERS.SIZE,

k. SUPPLIERS.SUPPLIER_ID,

l. SUPPLIERS.SUPPLIER_NAME

m. FROM CUSOMERS

n. INNER JOIN ORDERS ON ORDERS.CUSTOMER_ID = CUSTOMERS.CUSTOMER_ID

o. INNER JOIN PRODUCT_ORDERS ON PRODUCT_ORDERS.ORDER_ID = ORDERS.ORDER_ID

p. INNER JOIN PRODUCTS ON PRODUCTS.PRODUCT_ID = PRODUCT_ORDERS.PRODUCT_ID

q. INNER JOIN PRODUCT_SUPPLIERS ON PRODUCT_SUPPLIERS.PRODUCT_ID = PRODUCTS.PRODUCT_ID

r. INNER JOIN SUPPLIERS ON SUPPLIERS.SUPPLIER_ID = PRODUCT_SUPPLIERS.SUPPLIER_ID

37. Plaintiffs' request is specific to the EID. ERO does not typically perform any queries directly from the EID transactional database for reporting or analytics purposes. As explained above, the EID is a transactional database and therefore the process for querying the EID—especially in the way Plaintiffs request—will require ICE to create technically complex queries. The complexity of the queries would be two-fold. In addition to needing to define all the direct and indirect relationships between the data, ICE would also need to write queries which can perform complex calculations to replicate the ICE versus CBP encounter methodology which currently exists in the IIDS data warehouse only.

38. Plaintiffs state they "will not challenge the Agencies' decision to redact any free-format fields or, in other words, fields that contain potentially unique individualized content (including PII) entered into each cell, as distinct from standardized-content fields that allow only standardized entries (such as dates, times, numbers, codes, or categories)." Therefore, ERO would not include any data fields in our results which ICE could foresee would require individual line-by-line review for redactions, as this would take millions of hours of review. As explained below, identifying fields that ICE could reasonably foresee would require line-by-line review for redaction from among the 12,000 fields will take significant time, and then structuring the search to avoid the fields identified will also take significant time.

MY KNOWLEDGE, TRAINING, AND EXPERIENCE LEADS ME TO ESTIMATE IT WOULD TAKE A FULL TIME EMPLOYEE ALMOST FOUR YEARS TO DESIGN AND CONDUCT THE SEARCH PLAINTIFFS REQUEST—EVEN BEFORE ISSUES INEVITABLY ARISE THAT REQUIRE FURTHER WORK ONCE THE SEARCH IS BEING DESIGNED AND EXECUTED

39. For this request, based on my training, knowledge, and experience, I understand "directly linked" to mean that data can be searched for and produced using a direct join between two tables which have an established primary key/foreign key relationship.

40. For this request, based on my training, knowledge, and experience, I understand "indirectly linked" to mean that data can be pulled using an indirect relationship between two tables, where there is no established primary key/foreign key relationship, but instead one or more other tables may be used to bridge the gap between them through intermediary relationships or shared attributes.

41. ERO has no existing EID queries that would produce data fully responsive to either Part 1 or Part 2 of this request.

42. ERO does not utilize or have access to any reporting methodologies used by CBP to define its Nationwide Encounters Dataset. ERO defines encounters (ICE versus CBP) using its own reporting methodology which does not exist directly in the EID database and is calculated in the IIDS data warehouse only.

43. In order to preserve the linkage of records with direct or indirect relationships as requested by Plaintiffs, at this time I believe the outputs for Part 1 and Part 2 would each be provided in a single file with one row of data per record (Case Record for Part 1 and Encounter Record for Part 2). Each row of data would contain denormalized data from tables found to have a direct or indirect relationship. If a relationship results in multiple records, due to one-to-many relationships, the data will be de-duplicated to provide record which is determined to be most relevant. Using ICE's judgement and knowledge of the data and what would be reasonable under the circumstances, ICE may choose the isolate the earliest relevant record or the latest relevant record. Alternatively, ICE could decide to provide two sets of columns for the same data elements one containing the earliest and one containing the latest which would preserve the data linkage. Depending on the number of relationships, this process could likewise take significant analytical time.

44. ERO can currently identify potential direct relationships between tables. Initial research indicates there are approximately 30 direct relationships relative to Part 1 and there are approximately 130 direct relationships relative to Part 2. Even if a direct relationship does exist between two tables, there is analysis needed to determine how to utilize the relationship. For example, throughout the EID even where a direct relationship exists there will be varying levels of cardinality. These are typically One-to-Many relationships. Each One-to-Many relationship must be evaluated to determine if additional parameters should be written into the query to isolate a single record, such as earliest or latest, or whether all results should be returned. To fully analyze each potential direct relationship, approximately 2 hours per relationship is required. This effort would take approximately 320 hours total, or 60 hours for Part 1 and 260 hours for Part 2.

45. ERO cannot currently identify all indirect relationships between the tables in the EID which would be required to fulfill this request. To determine indirect relationships, we would look at each database table and attempt to work back to the tables containing Case (Part 1) and Encounter (Part 2) data using other tables and their primary and foreign key relationships. Each time another intermediate table is evaluated, the cardinality must also be evaluated to determine if we need to isolate a single record, such as earliest or latest, or whether all results should be returned. If multiple One-To-Many relationships exist in the intermediate tables, the query results will quickly increase exponentially, so when evaluating indirect relationships, it is typically advisable to isolate as many single records as possible. To fully analyze each potential indirect relationship, approximately 4 hours per relationship is required. Removing the 160 direct relationships mentioned previously, that leaves approximately 840 additional tables to analyze for an indirect relationship. If analysis of both Part 1 and Part 2 are performed at the same time, this would save some time, so instead of taking 8 hours (4 hours of analysis for Part 1 and 4 hours of

analysis for Part 2), it would take approximately 6 hours per relationship. This effort would take approximately 5,040 hours for Part 1 and Part 2 if performed at the same time, or 3,360 hours *each* if performed individually.

46. Tables which are found to have either a direct or indirect relationship relevant to Part 1 and Part 2 of the request would need to be reviewed to determine which columns should and should not be included in the query results based on which data fields would require a line-by-line analysis for redactions. This review would include evaluating each data field to determine which are entered via free-text data entry and would also identify which data fields house data that is exempt from disclosure, including but not limited to personally identifiable information and law enforcement sensitive information (among other types of information that is exempt from disclosure or that could lead to re-identification of an individual). If we assume that 75% of the tables in the EID can either be directly or indirectly linked relevant to Part 1 and/or Part 2 of the request (not every table in the EID will have a direct or indirect link back to the tables containing case and encounter data, but it is our reasonable estimate based on the characteristics of the database that the substantial majority will), then 750 tables would need to be analyzed to determine which fields to include and which to exclude. It is estimated that this analysis would take approximately 2 hours per table. This analysis would be performed simultaneously for both Part 1 and Part 2, so the total estimated time required is 1,500 hours.

47. When performing analysis of an encounter-based dataset, ERO relies on a calculated field in IIDS to allocate an encounter to either ICE or CBP. This calculation in IIDS is based on a reporting methodology maintained by ERO in all of our IIDS based datasets, so if ERO were to fulfill a request similar to Part 2 of this request, which requires identification of CBP encounters, we would rely on this standard methodology. Since this methodology does not exist as

an EID-based query, it would require extensive work to design, develop, and test. This methodology is complicated because it does a point-in-time analysis to determine the currently assigned duty location and agency of the arresting agent/officer at the time the arrest was performed. This query would involve several other queries, including obtaining the timestamp of the apprehension against the full history of the agent/officer's duty location history. The precise duty location history data is not currently available in EID. It is estimated that the effort to create EID functionality for this calculation would take approximately 500 hours.

48. Part 3 of this request is partially redundant with Parts 1 and 2. In the request for Parts 1 and 2, Plaintiffs request ICE pull all data with direct or indirect links back to the datasets Plaintiffs seek. In the fulfillment of these requests, tables containing code lookups would be denormalized directly back into the resultant datasets, so there is no need to provide them as stand-alone extracts. However, to the extent Plaintiffs seek "records that translate specific codes used in connection with the datapoints . . . into their corresponding meaning," beyond how those codes appear in the EID, ICE would need to expend additional resources deciding how to search for this information and make production of the information, which may involve producing additional documents or, in its discretion, ICE might decide to provide some or all responsive information in a data release.

49. Once the above analysis is complete, ERO would then need to build EID queries that would be run to search for this data. This task would involve using the results of the analysis described above to write code in the SQL programming language. The SQL code developed would be very complicated as it would potentially need to join up to 750 tables together using primary and foreign key relationships while isolating specific records due to cardinality constraints. Any SQL code developed would need to be thoroughly tested to ensure that the desired query results

are being produced. Once the query is developed and tested, ICE anticipates that execution of the query alone would take up to a week to process, and then a significant post-execution data validation process would need to be performed. ICE estimates the development, testing, and execution of the queries for Part 1 and Part 2 would take approximately 500 hours per request for a total of 1,000 hours.

50. The total time to analyze the database, develop the encounters methodology, build, and execute the queries for both requests is estimated to be 8,360 hours. Using a 2,080-hour work year, this request would require approximately 4 years of work for a single individual to complete.

51. I would like to note that to my knowledge fulfillment of these requests would be significantly more complex than any other data extract ever performed by ERO. The level of effort estimated in this declaration is based on my many years of experience working with data housed in the EID database. That being said, undertaking a project of this magnitude would come with significant risk. It is highly likely that unexpected challenges would arise—such as changes in requirements (or changes in our interpretation of the requirements), resource availability, or technical difficulties—that could significantly impact the amount of time to complete.

52. It is difficult to estimate the total size of the requests without performing the analysis described above. Our best estimate is that Part 1 would be approximately 2.5 Terabytes of data while Part 2 would be approximately 1.25 Terabytes of data. This equates to approximately 1,500,000,000 pages of printed text (assuming 2,500 bytes per page).

53. I do not have as much personal experience with the actual delivery of data to FOIA requesters, so the following information is based on my understanding of the process as explained to me by the ICE FOIA office. Data requests are typically delivered to FOIA requesters using the DHS SecureRelease Portal. There are no size limits of the SecureRelease Portal, but datasets this

large are not standard practice. It is estimated that it will take 20+ days to upload the 3.75 TB of data into the SecureRelease infrastructure. During this 20+ days, someone would need to be logged in and monitoring the upload, creating a burden to have someone available 24 hours to monitor this activity to ensure it does not fail. There is a standard process which runs to image the input files to convert them into tabular data which can be easily shared. This process would not be able to run on a dataset this large though, so custom development would be required to "turn-off" this functionality for this upload. It is unknown how long this would take. Once the data is loaded into the infrastructure, it will need to be transferred to the Requestor Portal and then downloaded by the requestor. It is unknown how long each of these two tasks will take. During the process, other users in the system could experience delays in using the system which would slow down the processing of tens of thousands of FOIA requests across multiple components.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct. Executed on the 1st of October 2024.

TIMOTHY P GIBNEY (affiliate)

Digitally signed by TIMOTHY P GIBNEY (affiliate)
Date: 2024.10.01 13:41:25 -04'00'

Timothy Gibney